NUMBER 43

JANUARY 1996

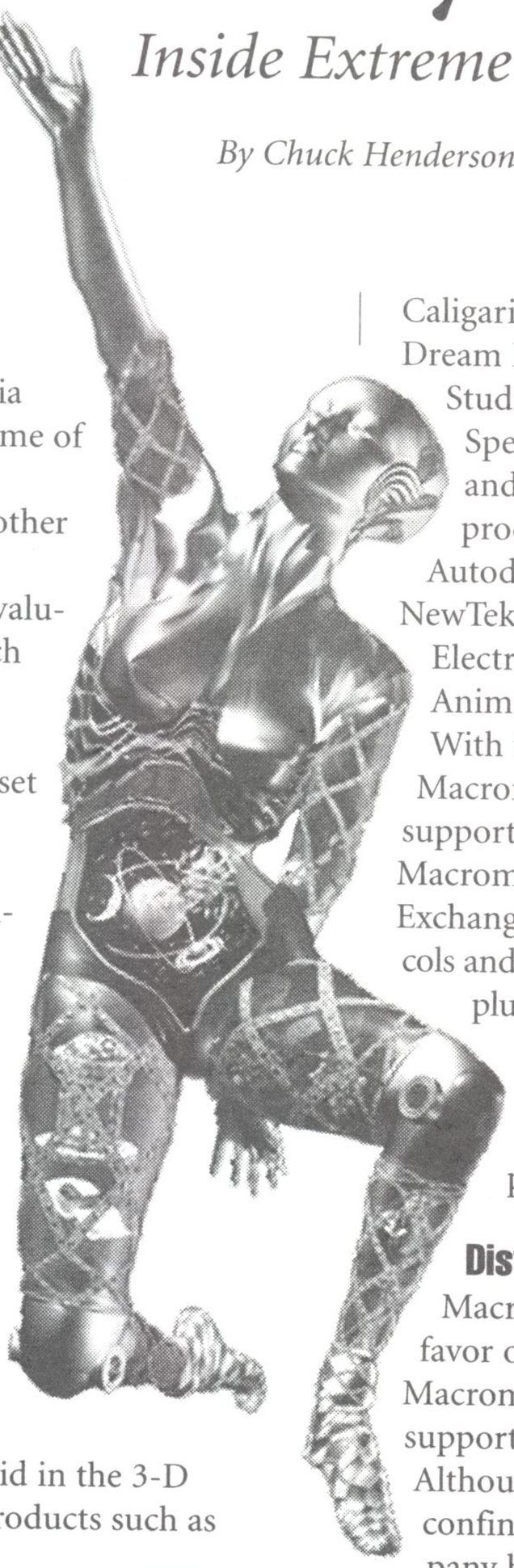
MUJ MACROMEDIA USER JOURNAL

http://www.hyperstand.com

Extremely Yours

Inside Extreme 3D

By Chuck Henderson



This incredible graphic, known affectionately as "Fiona" among Macromedia staffers, was created by Donald Grahme of Grahme Harding Productions using Extreme 3D (and a whole bunch of other tools). "Fiona" has been retained by Macromedia to invite the world to evaluate and experience Extreme 3D, which it hopes will bring a new level of affordable 3-D modeling, animation and rendering to the desktop. It was set to ship by February.

Users of either MacroModel or Macromedia Three-D will find familiar territory in Extreme 3D. Its spline-based surface modeling is the same as in MacroModel. The control of every last object parameter in the animation score is the same as in Macromedia Three-D. But the features that merge these two products in Extreme 3D will open up new creative opportunities for Macromedia devotees and those in pursuit of the right 3-D modeler, animator and renderer.

Extreme 3D is designed to fill a void in the 3-D market between low- to mid-range products such as Caligari TrueSpace, Ray Dream Designer, Strata Studio Pro and Specular Infini-D, and professional products such as Autodesk 3D Studio, NewTek Lightwave and Electric Image Animation System. With this and other Macromedia products supporting the Macromedia Information Exchange (MIX) protocols and the cross-product plug-in features of the Macromedia Open Architecture (MOA), the company is betting that Extreme 3D will leave Ray Dream Designer in the dust and make the entire family of integrated Macromedia products even more irresistible.

"Fiona" is from the *Body, Mind and Soul* animation created by Peter Spreenberg and Dennis Poon of IDEO Inc.

Distributed Rendering a Standard Feature

Macromedia has abandoned Pixar's RenderMan in favor of its own renderer. Or maybe Pixar abandoned Macromedia by announcing that Pixar will no longer support RenderMan on the Mac and Windows OS. Although Macromedia is asking Extreme 3D users to confine themselves to its proprietary renderer, the company has announced, as if to sweeten the bargain, that it will ship Extreme 3D with a distributed renderer.

The distributed rendering package will enable any multimedia shop with a multi-platform network to create

(continued on page 2)

IN THIS ISSUE...

2 **NEWS:** At Macworld, Macromedia focuses on SoundEdit and xRes.

7 **onLINGO:** Kurt Cagle examines Lingo at the dawn of Macromedia Common Scripting.

9 **AUTHORWARE:** Part 2 of Tom King's look at the Rapid ST prototyping and simulation tool.

14 **TIPS & TECHNIQUES:** Jonathan Adams introduces you to answer judging with Director.

16 **VERSION WATCH**

Publisher: HyperMedia Communications Inc. Not affiliated with Macromedia Inc. Macromedia is a trademark of Macromedia Inc.

MUJ News

New SoundEdit and xRes

At this month's Macworld Expo in San Francisco, Macromedia announced that SoundEdit 16 version 2 is shipping and a new version of xRes is due toward the end of the month. SoundEdit 16 version 2 for Mac and Power Mac is the first Macromedia application to include the new cross-application Macromedia User Interface, making it easier for new users to learn other programs in the application suite. The popular audio-processing and -editing program also supports the new Macromedia Open Architecture Xtras, so third-party developers can create special-effects filters and new extensions for the program. Major new features include batch processing of bit depth, compression, sample resolution and file format; and support for audio CD capture, IMA compression, and the .au file format for the Internet.

XRes 2.0 is the first Macromedia version of the image editing program that came with the Fauve acquisition. The strength of xRes has always been its ability to handle image files as large as several hundred megabytes in real time when working in xRes mode. In that mode, the program only processes the visible area of an image at the current zoom level, and waits to process the entire image during the final-rendering stage, after all work is complete. This makes the RAM requirement for huge files significantly less than when working with every pixel of an image throughout the editing process, as with other image editing programs such as Adobe Photoshop.

XRes 2.0 has lots of new features that build on this core strength, including the Macromedia User Interface look and feel. A new Direct mode allows users to work at the traditional pixel level, as you would with Photoshop. In addition, Photoshop filters are supported while working in the Direct mode. Indexed Color mode allows users to work in 8-bit color for screen-destined work. In this mode, users can set a Matte color before importing the work into Director and compositing it on Director's Stage. For images destined for the Internet, xRes supports GIF89, Progressive JPEG and the new PNG formats. The program supports batch processing of images to and from its native MMI file format.

Image editing enhancements include multiple undos, new path functionality and support for FreeHand paths, 20 new pressure-sensitive brushes, multi-object editing and object-oriented compositing—all image elements and text are preserved as discrete elements, with their own 5-bit alpha channels and undo paths.

XRes 2.0 for Mac, Power Mac and Windows will be included in the Macromedia bundles FreeHand Graphics Studio, and Director Studio will be available separately for $699 ($99 for previous xRes users, or $199 for users of Adobe Photoshop, Live Picture or Fractal Design Painter).

Macromedia Inc. (415) 252-2000;
http://www.macromedia.com

(EXTREMELY YOURS: from page 1)

animation frames from within Extreme 3D on each of its platforms simultaneously—that means Mac and Windows machines rendering together, provided you have one copy of Extreme 3D for each platform. If you have 20 machines that are idle at night, for example, you can create an animation sequence as a sequentially numbered group of PICT files that can be rendered on whichever machine is available at the time. While machine A renders Frame 1,

MUJ MACROMEDIA USER JOURNAL

http://www.hyperstand.com

EDITORIAL DIRECTOR: Phil Hood
EXECUTIVE EDITORS: Tony Bové and Cheryl Rhodes
MANAGING EDITOR: Eric Elia
CONTRIBUTING EDITORS: Jonathan Adams, David Biedny, Kurt Cagle, Gary Gattis, Chuck Henderson, Tom King, Lynda Weinman
COPY EDITOR: Pat Soberanis
ART DIRECTOR: Nancy Cutler
ASSISTANT ART DIRECTOR: Jake Flaherty
ART INTERN: Agnes Chan
PUBLISHER: Penny Ertelt
ASSOCIATE CIRCULATION MANAGER: Frank Ha

Subscriptions are $175 ($200 outside the U.S.) for 12 issues, $250 ($275 outside the U.S.) for 24 issues. Back issues are available upon request. Single copies are $12, or $6.50 each for 10 copies or more.
Direct all subscriptions, sample copy and back-issue requests to: *Macromedia User Journal*, P.O. Box 469079, Escondido, CA 92046-9779. Voice: (800) 546-6707 (9:00 a.m.–5 p.m. PST) Outside the U.S., call (415) 573-5170, ext. 249. Fax: (619) 743-2590 MUJSub@newmedia.com.
Send article submissions and letters to: *Macromedia User Journal*, 901 Mariner's Island Blvd. #365, San Mateo, CA 94404. MCI Mail and AppleLink: newmediamag; Internet: MUJEditor@newmedia.com.
Tony Bové: 30777 S. Hwy, 1/P.O. Box 1289, Gualala, CA 95445. MCI Mail: TBOVE; CompuServe: 70105,722; AppleLink: D0493; America Online: TonyBove; Internet: tony@rockument.com.
Macromedia Director, Macromedia, MediaMaker, Macromedia Three-D, Lingo, Windows Player and XObjects are trademarks of Macromedia Inc.
This publication is not affiliated with Macromedia Inc., nor is Macromedia responsible for its contents.

Macromedia User Journal is published 12 times a year by HyperMedia Communications Inc. Copyright ©1995 by HyperMedia Communications Inc.
(ISSN 1065-3929)

Got a question about your* Macromedia User Journal *subscription?
Call (800) 546-6707 for questions regarding billing and delivery.

machine B renders Frame 2, machine C renders Frame 3, and so on. If your computers take five minutes to render each frame and you have 20 computers available for 10 hours at night, you can renderer 2,400 frames overnight, come into the office in the morning, jam the frames into a Director movie or Adobe Premiere, and convert those 2,400 PICT files into an 80-second, 30fps QuickTime animation. If you keep that up for 30 days, you will have generated 40 minutes' worth of animation. Macromedia will ship the distributed renderer as a standard part of a single-user package of Extreme 3D.

Big-Time Texture Mapping

Of the 16 new features that show up in the Extreme 3D tool palette (see Figure 2), five are dedicated to the task of texture mapping. Texture mapping is the eminently useful process of pasting a bitmap (e.g., a PICT file) on to a 3-D surface. Texture mapping is similar to applying a decal or attaching a label to a bottle or other 3-D object. But the power of texture mapping goes well beyond replicating what you can do in the material world. With texture maps you can, for instance, cover an entire complex surface with a landscape of your own making, or with a repeating set of images that tile together seamlessly. And the tool-palette selections that address texture mapping make this process remarkably simple.

Texture maps (PICT files) can be edited with the Extreme 3D Material Browser, a new tool that, like the object and view browsers in MacroModel, allows the user to manipulate a variety of parameters for a given material, in this case a texture map. It is with the Material Browser that texture maps are selected and applied to the surface of an object. Alpha map controls allow complex texture maps with irregular borders and holes to be imported into the model as well.

The Intrinsic Texture Placement tool allows you to manipulate the placement of a texture map. The term *intrinsic* means that the texture map is a "smart" object that can be manipulated independently from other objects. Therefore, like any other object, the texture map, once placed on a surface, can be moved, scaled, rotated and tiled so that it appears to "flow" over the surface. For example, by selecting the texture and holding down the F2 or Control key, the texture can be scaled in real time on the 3-D object to which it has been applied. If you scale the texture map to a size much smaller than the 3-D object, you can click on the texture and interactively move it around the object until you find the position you want.

By invoking the Object Browser, you can fine-tune many additional parameters that determine how the texture map interacts with a surface. This includes a tiling function that paints the texture map repeatedly over the entire 3-D surface, and horizontal- and vertical-flip functions that rotate adjacent tiles along their common axis. The flip functions are very useful in creating complex textures from small files so that they appear as a seamless texture over a large or complex surface.

The Cylindrical Texture Placement tool works in much the same way as the Intrinsic Texture Placement tool. But in the case of mapping to a cylinder, we are concerned with the texture map's length and the angle of rotation. When pasting a label on a bottle, for example, the height of the label needs to be adjusted; its placement on the bottle needs to be determined; and the angle the label occupies around the bottle (usually about 120 degrees) needs to be assigned. The Cylindrical Texture Placement tool allows for height, placement or angle to be manipulated by holding down the Control key, no key or the Option key, respectively, while clicking on the texture object.

A similar tool, called the Spherical Texture Placement tool, allows you to perform manipulations of the texture map, in which the surface object is treated like a sphere.

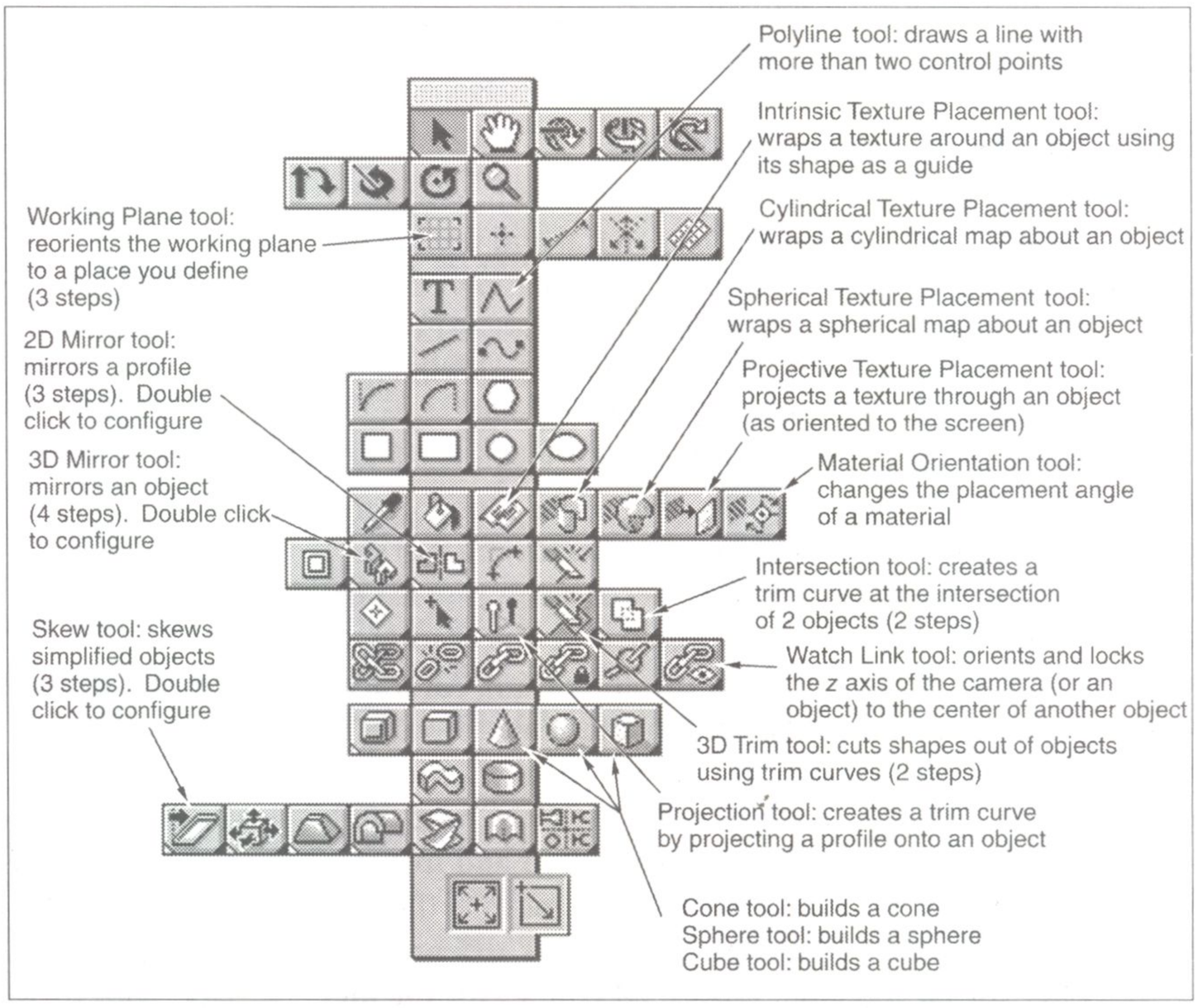

Polyline tool: draws a line with more than two control points
Intrinsic Texture Placement tool: wraps a texture around an object using its shape as a guide
Cylindrical Texture Placement tool: wraps a cylindrical map about an object
Working Plane tool: reorients the working plane to a place you define (3 steps)
Spherical Texture Placement tool: wraps a spherical map about an object
2D Mirror tool: mirrors a profile (3 steps). Double click to configure
Projective Texture Placement tool: projects a texture through an object (as oriented to the screen)
3D Mirror tool: mirrors an object (4 steps). Double click to configure
Material Orientation tool: changes the placement angle of a material
Intersection tool: creates a trim curve at the intersection of 2 objects (2 steps)
Skew tool: skews simplified objects (3 steps). Double click to configure
Watch Link tool: orients and locks the z axis of the camera (or an object) to the center of another object
3D Trim tool: cuts shapes out of objects using trim curves (2 steps)
Projection tool: creates a trim curve by projecting a profile onto an object
Cone tool: builds a cone
Sphere tool: builds a sphere
Cube tool: builds a cube

Figure 2: The exploded Extreme 3D (v1.0.0 b3) tool palette, showing new tools.

The Projective Texture Placement tool applies a projection of the texture across all selected objects. By grouping several objects in proximity to one another, selecting a texture material, and then invoking the Projective Texture Placement tool, a texture map can be applied to all the projected objects simultaneously, as if a slide projector were being used to project an image on all of the surfaces. After the projected texture map is applied, all of the objects can be moved, rotated, linked or otherwise displaced, and they will each retain the portion of the texture map they received during the projective texture map placement. Extreme 3D is one of the only desktop 3-D editors that allow you to apply textures to multiple surface objects in such an interactive editing environment.

All the texture functions can be manipulated over time and animated just like a surface object. A texture map can change position, shrink or rotate onto a different facet on the 3-D surface as an animation plays. The texture is an object itself that is linked to a surface object(s). If a surface moves in 3-D space or changes its own geometric configuration (such as a blooming flower), the texture object, and any animated changes to the texture map, will likewise change with the surface object. Parameters that define the texture object itself can also be scored to change over time. Finally, parameters such as opacity can also change as a function of time.

This level of dexterity is very attractive to those of us who have labored over the problems in texture-mapping animation using other rendering tools.

If You Can Edit It, You Can Animate It

In Extreme 3D, each editable parameter—from model geometry and light parameters to material and texture parameters—can be animated. Even deformation operations can be animated with associated key frames in the score. What results is a completely plastic animation environment that can be controlled frame by frame.

The score is a time-based database that keeps track of everything that takes place in the animation. Horizontally, the score represents time or, more specifically, frames in the animation. Vertically, the score keeps a running list of all the tracks that apply to each object, and object subparameter, that are participating in the animation. All tracks of all objects can be visualized and manipulated from within the score for any discrete time interval (frame number). Because even a simple animation can have literally hundreds of tracks that can be visualized and manipulated from within the score, Extreme 3D permits the separate viewing of active tracks, position tracks, orientation tracks, scale tracks and color tracks. Of course, for the diehards, all tracks can also be viewed together.

Mixed Render Styles

While constructing animations, we are used to working in either wireframe mode or real-time rendering mode. With Extreme 3D, we can mix modes by assigning different rendering styles to separate objects in the animation. So the leaves on a flower stem, for example, can be fully rendered in color while the flower is rendered in wireframe mode. Objects can even be quick-rendered in box mode, where a simple box represents the entire object. Box mode is particularly useful for objects that are complex and thus take a long time to render even in wireframe mode. Also, when one object is buried inside another, by setting the outer object in box mode, the inner object becomes visible and easily selectable and modifiable.

Boolean's Back Door: A Mini-Tutorial

Boolean operands in the domain of 3-D solid modeling refer to the ability to subtract (or add) one 3-D object from (or to) another. For example, by subtracting a cylinder from a sphere you can, by using Boolean operands, create a perfect hole penetrating all the way through the sphere.

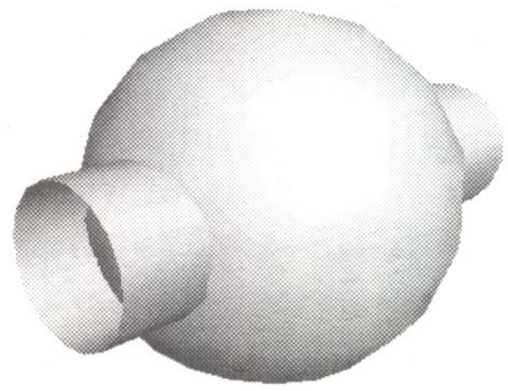


Figure 3: A cylinder bisecting a sphere.

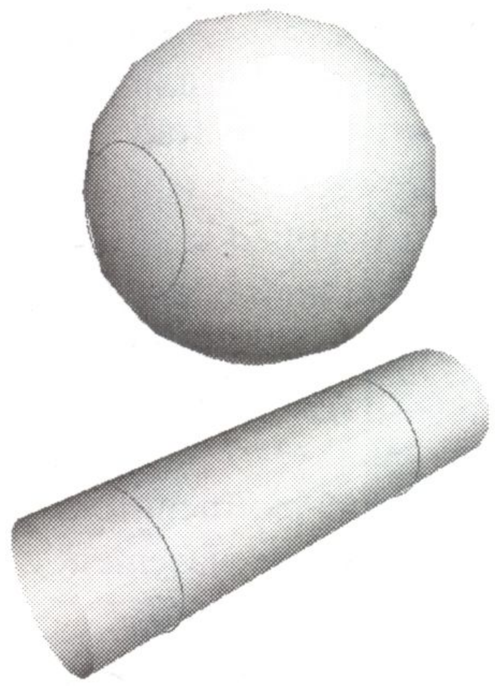


Figure 4: Shapes with intersection lines.

Extreme 3D is not a solid modeler but rather a surface modeler (e.g., a cylinder is represented by a curved surface, with or without a cap on each end). Creating the effect of a cylinder being subtracted from a sphere

involves a few more steps than a simple subtract command. First, the cylinder must be positioned within the sphere. Second, the intersection of the cylinder and sphere must be identified as a "trim surface." Third, the cylinder and the sphere must be separated so they can be operated upon independently. Fourth, the ends of the cylinder and the two intersection holes in the sphere must be deleted. And finally, the remaining section of the cylinder must be precisely repositioned inside the sphere.

Let's look at this procedure a little more closely.

Figure 3 shows a cylinder without end caps bisecting a sphere. There are two intersection lines; together they make up a complex curve that rides the edge of the cylinder and the sphere where the two surfaces meet.

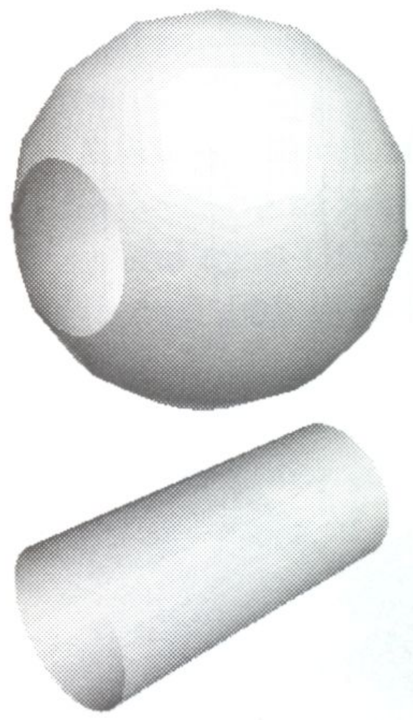


Figure 5: Shapes without end sections.

By selecting the intersection tool and clicking first on the cylinder near the intersection, and then on the sphere near the intersection, Extreme 3D goes about the business of precisely locating the intersection lines and highlighting them on the screen, as seen in Figure 4.

By viewing the model directly from the front (or from another cardinal position), the cylinder is moved away from the sphere so the trim surfaces can be removed. The trick here is to select the cylinder, open the object browser, copy the *x* (horizontal) value to the Clipboard and then move the cylinder away from the sphere along the *x* axis by holding down the H key. Holding down the H key restricts movement along the *x* axis. After moving the cylinder, the intersection lines show up on both the cylinder and the sphere.

Next, select the 3D trim tool and click on one of the intersection lines of the cylinder. Carefully go to the other intersection line and, while holding down the Shift key, select the other intersection line on the cylinder. Now click on one of the end sections of the cylinder (the parts we want to trim off) and both end sections should disappear, as shown in Figure 5, leaving only the center section.

The procedure for the sphere is only slightly different. Use the 3D trim tool again, only this time click on the intersection line on the sphere. You probably can see only one, so we'll do one at a time. Now click on the interior of the intersection line (the part of the sphere we want to get rid of) and it should disappear. Reverse the view of the

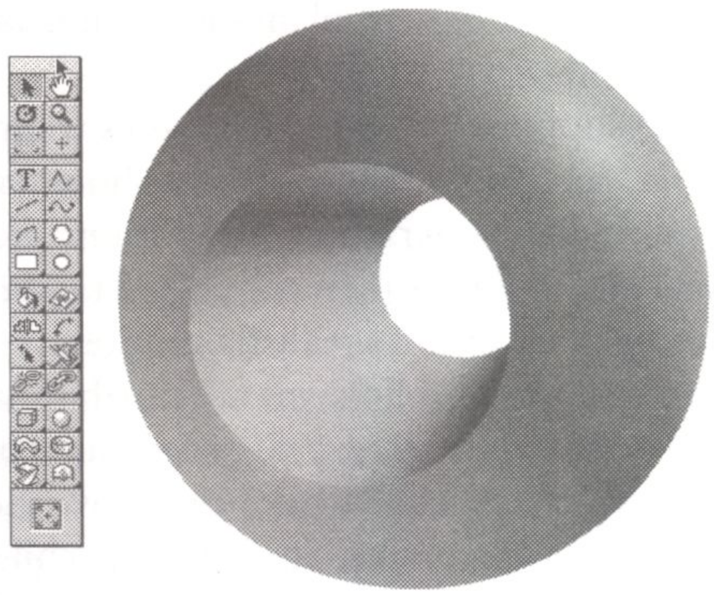


Figure 6: The finished Boolean shape.

sphere so you can see the other intersection line and repeat the procedure, getting rid of the other circular patch on the cylinder.

Now for the fun part. Select the cylinder section and, with the object browser still open, paste the *x* value you saved to the Clipboard into the *x* value of the object. The cylinder section should immediately pop back into the sphere precisely aligned, so that the edges of the cylinder section meet exactly with the hole through the sphere, as seen in Figure 6.

Render to taste and voilà, a Boolean operand. I have used this technique on a cylinder that is only partially intersecting the sphere, so that the intersection line is a very complex saddle shape. Guess what? Extreme 3D didn't break and the Boolean intersection was just as easy to complete as when the cylinder passes directly through the sphere.

Forward (But Not Inverse) Kinematics

Inverse kinematics (IK) refers to how linked objects are manipulated. Suppose there is a hierarchy of a shoulder, an elbow and a wrist. To have the hand reach something using inverse kinematics, you would grab the wrist and the upper joints would respond in what is supposed to be a realistic motion. In practice, major development efforts are required to produce IK systems that avoid joints bending at impossible angles. Each joint needs to be carefully described so that the permitted ranges of motion around the *x*, *y* and *z* axes are known. Further, joint priorities need to be specified. If the wrist is moved, is most of the motion in the elbow or the shoulder?

For sophisticated computer baseball games, for example, designers spend thousands of dollars photographing the swings of famous players so that they can reverse-engineer the inverse kinematics of their movements. It has been found that inverse kinematics usually requires that object movements be confined to a two-dimensional plane rather than be free-wheeling within the 3-D world. Isolating these 2-D movement planes is a major task in building IK systems.

Extreme 3D supports forward kinematics where, in the above example, you would first move the shoulder, then the elbow and finally the wrist. Each key frame created using forward kinematics is then connected by spline-based "tweening" that mimics realistic motion.

Extreme 3D's joints and linkages are much more powerful than those found in many 3-D programs. There are "free links," in which the child object can move independently of its parent; "lock links," in which the child object is fixed on its parent and the two cannot move independently; "ball links," in which the position of the child object is fixed vis-à-vis its parent at only one end; and "watch links," which allow an object, light or camera to track another object over time. Any link can be restricted to any desired degree of freedom in the object browser. Further, the use of construction axes for linkages allows for confinement of movement by an arbitrary geometry.

Real Bézier Curves

Extreme 3D supports Bézier curves (rather than B-splines)—the same type of curves used in FreeHand. Many 3-D modelers boast spline-based modeling but really have only spline design tools. In these products the underlying data is really static polygons. Some modelers, such as the plug-ins for 3D Studio, deal in single spline patches only. In Extreme 3D, all the information is stored as webs of real, interconnected Bézier splines. Although the wireframe display is in polygons, the underlying database in Extreme 3D defaults to Bézier splines, which allow smooth shapes. They are more intuitive because the control points are right on the curve, so you can directly manipulate the surface. Also, Bézier spline control points can be at angles to each other, which lets you make sharp edges or very tight curves.

Lights

The lights in Extreme 3D can be made to be visible in space. And, just like a movie projector would be in a smoky movie theater, the cone of light extending from a spotlight can be made visible via the "dust effect." With the dust effect, a spotlight can simulate a lens flare, although lens-flare "artifacts" will not be seen and will need to be added in post-production. Lighting effects that simulate shadows of tree branches or venetian blinds, etc. are not yet supported by Extreme 3D.

They Fixed It!

Several months ago I heard that one of the most vexing problems with MacroModel would be fixed with Extreme 3D—namely the problem of skinning profiles when each has a different number of control points. In the past, because of the control points and because the start or "home" position of one profile might not line up with the next, the resulting skin often became a twisted knot between adjacent profiles. There was a workaround, but it wasn't pretty.

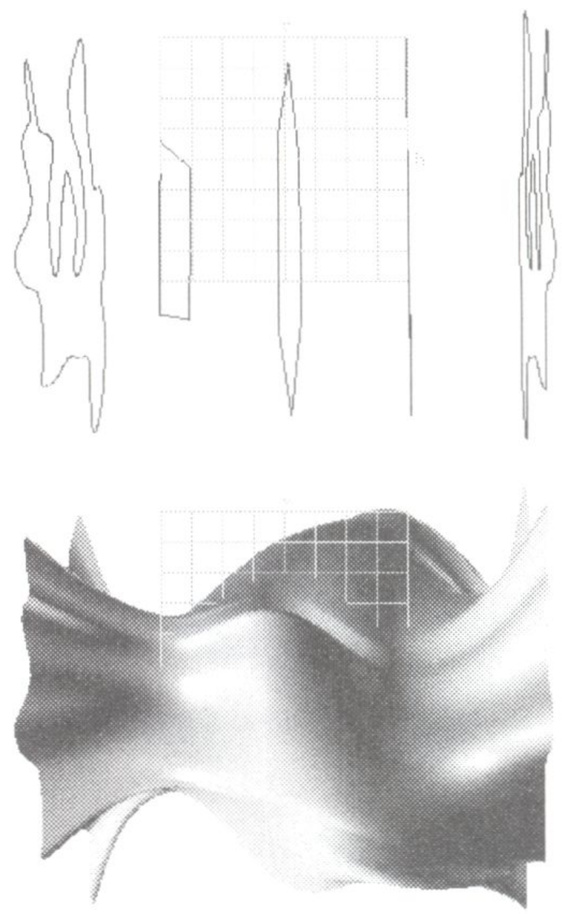


Figure 7: The skin tool can now produce logical skin surfaces without worrying about the control points or start positions of participating profiles.

To my great surprise and pleasure, Extreme 3D vanquishes the skin demons once and for all. I have built several skins using Extreme 3D. One was made up of five profiles; the first had about 16 control points, the second just three, the third had four, the fourth had six and the last had 16 again. After running the skin tool, the resulting shape was very organic, and it did basically what I had hoped for—intersected the profiles without contorting the skin (see Figure 7).

There's a lot more to talk about concerning the Extreme 3D offering. Look in upcoming issues for a series of articles and tutorials about how to get work done quickly and efficiently using this new, powerful multimedia tool.

Chuck Henderson (earthmap@mcn.org) is vice president of research and development for XYZT Corp. in Gualala,

onLingo

Archetype and Architecture

Open Architecture and the Future of Lingo

By Kurt Cagle

A term has come into vogue in the programming community: open architecture. Its fundamental principles imply constructing your operating system so that it becomes machine independent—the same source code will run similar programs regardless of whether that program is running on a 68000 processor, a Pentium Pro, a UNIX box, whatever. It's a faintly heretical notion to many; the devotion of people to the Macintosh is legendary, and most UNIX users will only sneer when they have to deal with something touched by Microsoft and Intel.

However, this particular notion is driven by both capability and a necessity. The capability comes from the increasingly interconnected nature of computers. Back when Windows first crawled out of the sea, and the Macintosh was more memorable for the woman swinging her hammer through Big Brother's face than it was for its OS, the idea that most machines would be talking to one another would have been ludicrous: Their architectures were too dissimilar. Today, most corporate computers (Mac, Windows and UNIX) are now attached by local area networks, and many are increasingly talking to other LANs through wide area networks—the most notable of these WANs being, of course, the Internet. Thus, it's becoming increasingly desirable to be able to run your spreadsheet program, your interactive product demo or your killer computer game on all the types of machines out there. In order to do that, they must be able not only to send signals back and forth, but to have an agreed-upon standard by which to work.

Macromedia has recognized this principle, and one can say to a certain extent that its architectural model for Director 4.0 was built upon it. Director (and hence Lingo) makes use of the concept of the virtual machine to be able to port code from one machine to another. In principle, the idea is simple: When you create a Director application, the code that you write communicates with a set of APIs (application programming interfaces) built deep into the heart of Director. These APIs take the generic calls generated by the code—draw a rectangle here, image a sprite there, fill this text cast member with this text—and then translate them into the dialect of the machine it resides upon. The upper-level code talks to the lower-level code through an intermediary layer known as the interface. This interface is not like the visual interface through which you to interact with your program, but its function is similar; the code-to-code interface acts as a translator between the upper-level language and the lower-level language, that is, between Director and the computer.

At the recent Macromedia User Conference, the star of the show was definitely Shockwave, (December 1995 *MUJ*). (I plan to discuss Shockwave's Lingo in much greater detail next month). Although Shockwave is certainly important, perhaps more significant, both for the company and for the development community, was the announcement that Macromedia is creating an open architecture for its entire product line. Macromedia Open Architecture (or MOA) is a vital part of Macromedia's survival and growth, and the implications for developers—and I'm not just talking multimedia developers—is profound.

Director suffers from a great number of shortcomings, not least of which is that it is a world unto itself. Its ability to communicate with the "outside world" is extremely limited, accomplished primarily through the mechanism of external-code entities such as the Serial Port XObject. Editing graphics has to be done either through Director's primitive graphics editor (which was anemic in version 3.0 and extremely pathetic by the time it rematerialized, virtually unchanged, in version 4.0) or by importing graphics from other applications and crossing your fingers. You can't import formatted text—hell, you can't import text, period, without standing on your head and reciting some of the more arcane Lingo mantras. Communicate data with another program? Better hope that the Clipboard isn't being used by someone else—its your *only* connection to the operating environment. Granted, you do have Lingo commands like ImportFileInto, but even the dedicated engineers at Macromedia will tell you that that command is primarily for the use of the developer, not the end user.

MOA changes all that. MOA is built upon the same foundation as Microsoft's OLE interface: the Component Object Model (COM). In this model, applications are no longer huge, monolithic entities that resemble giant department stores—that is, the store offers great selections in areas that may not be relevant to you, but next to nothing if you're looking for a special book or an off-size pair of pants. Instead, COM-like architectures can be thought of as malls—lots of little specialty shops, each providing great selections in their niche. If a given store isn't making a lot of sales (i.e., it's not being used very much), it can be jettisoned and a new store brought in. Under MOA,

Director can farm out its tasks, large and small, to different applications that share the MOA interface.

Say I'm working in Director on my notebook PC when I realize that I need an important graphic that is available on my Macintosh server at the office, and it's in a format that Director won't recognize. Today, I'd have to close out of Director, connect via a phone line to the office, start up my paint editor on the other machine, edit the graphic, save it in the correct format, close the paint editor, copy the file over to my machine, then import it into Director. If I discover I've made a mistake with the graphic, I have to repeat the whole process all.

With MOA, if I'm in Director 5.0, I can open up the Paint window and Director detects that the file is listed as being related to a different program. No problem. Director dials up the other computer automatically, locates the file, determines its type and then launches the MOA enabled editor on the Mac server. When I'm done, I close the window and discover an error in the graphic. No biggie. I open up the Paint window again (it should still be linked, since I haven't told it to shut down), make new changes, and I'm done. Media ends up residing on only one place (the server), which minimizes version-control problems.

MOA is significant for tool developers as well. I doubt there is a person who's needed database support who can't believe how much of a pain it is to incorporate data controls. Lists are powerful constructs, but they are messy to work with and not terribly fast. Third-party database tools are finally coming into their own, but they don't really have a lot of power and suffer from having to comply to a set of standards that are now pushing four years old. With MOA, developers can create database applications that are machine independent and that can be integrated seamlessly into the Director environment. Such tools are called Xtras, and they are rather remarkable.

An Xtra can be a full-blown program (such as a database application), a filter (such as one demonstrated at the conference that allowed a user to convert a text block into a 3-D–rendered version of the words), an interface (such as the aforementioned widget to connect you to your office) or a programming utility (such as one that allows you to see class relationships, which is vital for creating more complex class structures). Xtras would, in appropriate cases, work across platforms and across applications—the same 3-D filter would work in both FreeHand and Director, for example.

Macromedia, which for so long has been rather lacking in the planning department, seems to finally be getting its act together. The company will be making the MOA architecture (including Xtra specifications) available free to developers, thus seeding a significant market in third-party Xtra development. Hopefully, third-party activity will rival the proliferation of VBXs and OCXs that came with the introduction of Microsoft's Visual Basic. This in turn will make Director a viable development platform, especially since version 4.0 is beginning to look a little gray around the temples.

MOA and Lingo

All right, you say, that's all fine and dandy, but what does this have to do with Lingo? There's been a lot of fear on the part of developers that Lingo will transform radically from the current to the next incarnation of Director, rendering all of their programming moot (and how many of you still have closets full of unusable factory objects from 3.0?). The short-term answer is that Lingo will not change significantly between Director 4.0 and Director 5.0; indeed, a lot of the changes that are coming in the rest of Director are due to the frog-leap that Lingo took over the rest of the product with version 4.0. Lingo should remain quite stable this time around. Code Xtras will be incorporated using a mechanism that is very similar to the birth method of current objects, and the Xtras will then work in a manner similar to current parent script classes. A few things might get tweaked, and you can expect support for text in Lingo to improve dramatically.

Shockwave 5.0, which will ship concurrently with Director, will utilize the same MOA structure as the other Macromedia products, and the Lingo support for Shockwave will improve dramatically as a consequence. This means you'll be able to create that interactive showpiece for your Internet Web page that will also be able to transmit pertinent user data back to the Web server and build a real-time, fully interactive graphical MUD (all right!); with Shockwave, you'll become a player on the Information Uberbahn.

Lingo is a powerful language, though it didn't dawn on me until the conference how powerful it could be. Deep in the underground labs of Macromedia's most diabolical mad scientists, a new creature is being constructed. It's mother: Lingo. Dubbed Common Lingo or Macromedia Common Scripting (MCS), this new language will give users the ability to communicate at a higher level between various MOA-enabled applications. Do you need to download and batch-process a selection of newswire stories from the Internet every night, converting the headlines into stunning graphics while you get a good night's sleep? Well, MCS can do it. Want a user working with a FreeHand or xRes graphic to add a self-contained animation engine? MCS can do that too. If MOA is the mechanism that enables the applications to communicate with one another, MCS will be the interface that enables users to interact with the objects that they create. In time, Director itself might completely disappear (or revert back to its original purpose as an animation layout tool), while

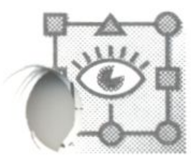
Lingo metamorphosizes into this butterfly called MCS—each object would then have its own animations, its own database capabilities, its own communications protocols. MCS will probably not become a reality with 5.0, but by Director 6.0, the world will hardly recognize the humble origins from which this new technology sprang.

Kurt Cagle is a freelance writer and software developer, and principal of Cagle Communications, in Bellevue, Washington, specializing in multimedia design and development. He can be contacted via e-mail at cagleK@aol.com, cagles@blarg.net, or by phone at (206) 557-8532.

Authorware

Rapid Returns

By Tom King



Last month we looked at Emultek's Rapid ST, an object-oriented simulation and prototyping tool based on state machines (a model for describing simulations). It runs under Windows and can work in cooperation with Authorware or independently. This month's column demonstrates how to use Rapid within Authorware.

Modeling Recap

I'd like to clarify state machines vs. traditional procedural (conditional) programming. Modeling in Rapid ST requires you to think a little differently. Instead of thinking of all kinds of conditions that trigger changes (e.g., if

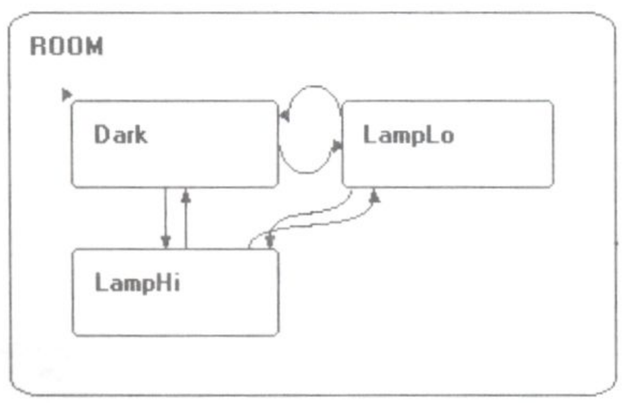


Figure 1: Logic chart based on lamp states.

the power is on and the lamp switch is on and the wall switch is on, then turn on the light), a good Rapid author thinks in terms of modes or states. Ideally, a Rapid program will have lots of modes, but very few conditions for triggering switches between modes. Why? Let's take a look at last month's lamp example.

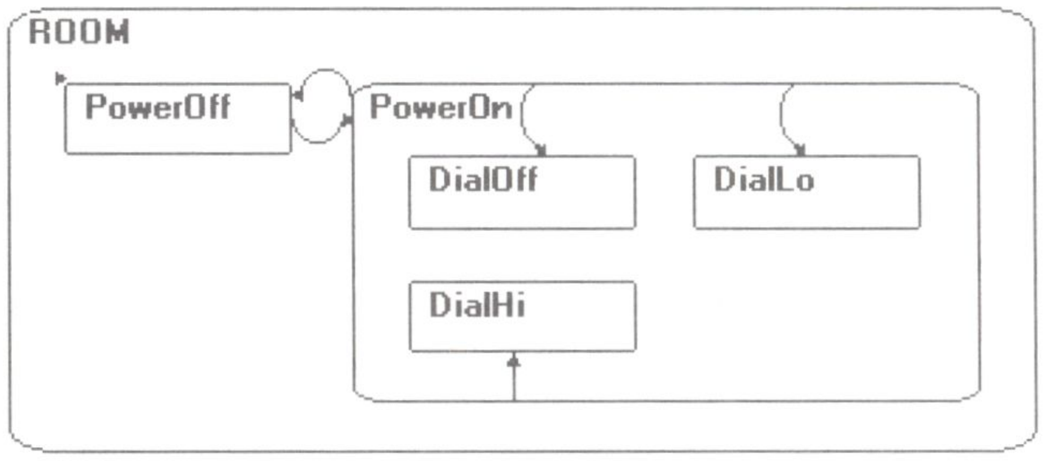


Figure 2: Logic chart based on switch states.

At first it may seem like Figure 2 is more complicated, since it has six modes vs. only four modes in Figure 1. In fact, Figure 2 is better for two reasons. First, having more modes is more explicit—it tells us more about the "machine" without even looking at it. For example, by looking at Figure 1, you can't tell if there is one switch or two, whereas Figure 2 conveys that information clearly. Second, having more modes reduces the complexity of the transitions.

You can tell that your Rapid program is going awry if you start developing very complex, conditional triggers. This can be very tempting if you start with a model like Figure 1, or if you slip into a traditional programming mind set and try to do everything based on if-then-else clauses. Examine Table 1:

Table 1: Logic Tree and Transitions Based on Lamp State

Transition From	Transition To	Trigger (in Rapid syntax)
{Default/Entry}	Dark	{start-up}
Dark	LampLo	WallSwitch.up is connected and Dial.Low is connected
Dark	LampHi	WallSwitch.up is connected and Dial.High is connected
LampLo	Dark	WallSwitch.down is connected or Dial.Off is connected
LampLo	LampHi	WallSwitch.up is connected and Dial.High is connected
LampHi	Dark	WallSwitch.down is connected
LampHi	LampHi	WallSwitch.up is connected and Dial.Low is connected

Look at the complexity of the triggers in Table 1, even in this simple model. All those trigger conditions can be a problem because they quickly propagate throughout your model. The *and*s and *or*s have to be watched carefully and require extremely conscientious maintenance during editing (or debugging). Compare the trigger conditions in

Table 1 with those in Table 2. In Table 2, notice how the child modes of PowerOn "inherit" its triggers, and therefore our Dial modes can ignore the WallSwitch.up status. Likewise, the transition from PowerOn to PowerOff applies to all the children, so they don't need the individual power off transitions included in Table 1. Don't forget, the whole reason we got started with this tool was to make quick changes to simulations—without tearing our hair out.

Table 2: Logic Tree and Transitions Based on Switch States

Transition From	Transition To	Trigger (in Rapid Syntax)
{Default/Entry}	PowerOff	{start-up}
PowerOff	PowerOn	WallSwitch.up is connected
PowerOn	DialOff	Dial.Off is connected
PowerOn	DialLow	Dial.Low is connected
PowerOn	DialHigh	Dial.High is connected
PowerOn	PowerOff	WallSwitch.down is connected

Free and Easy

Once you've developed an understanding of modes and transitions, developing a full, free-play simulation is easy. You may recognize some of the following steps from the high-level description in last month's column. First, drag the objects into position using the Layout Editor. Figure 3 shows a layout in progress for the lamp simulation. From the left column of the palette I've selected the switch object, and from the right column I've selected the type of

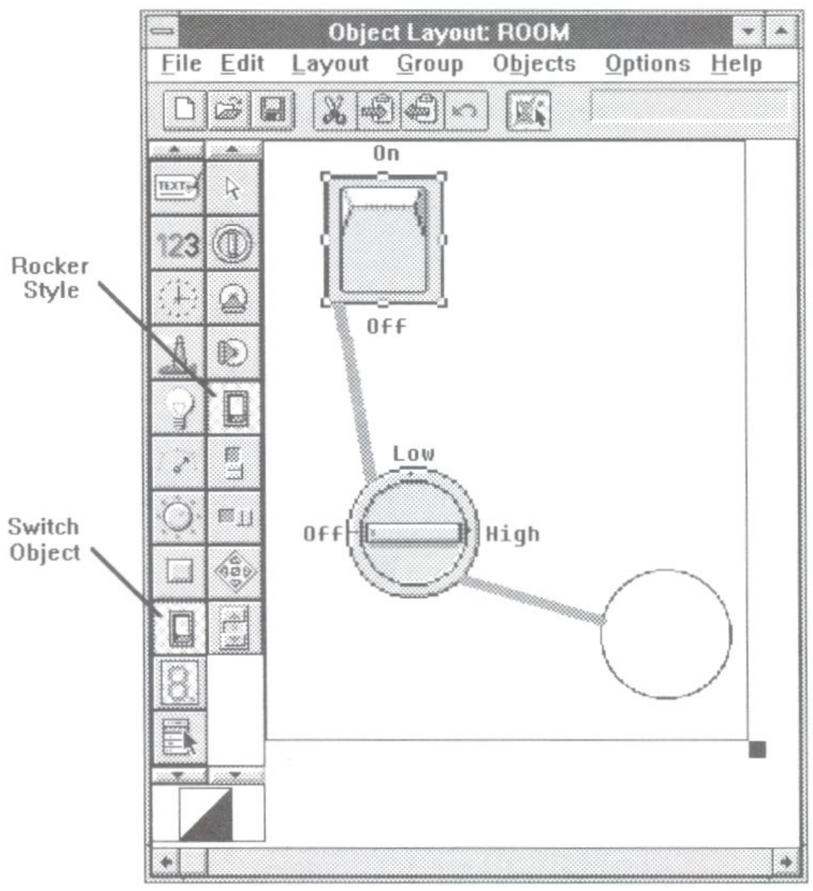


Figure 3: Creating a switch.

switch I want—a rocker switch. Dragging a selection marquee creates the switch, as shown in Figure 3.

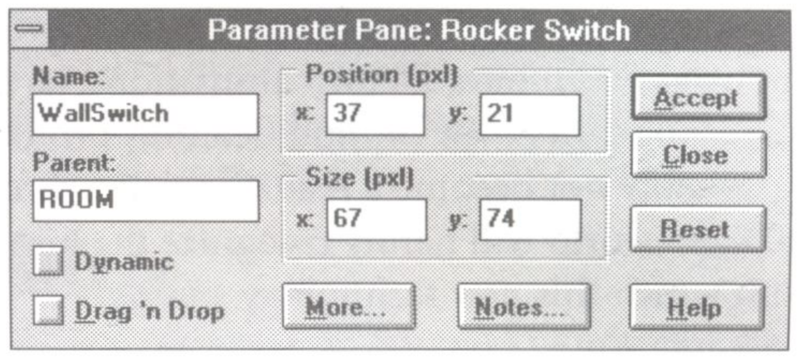


Figure 4. Naming the switch.

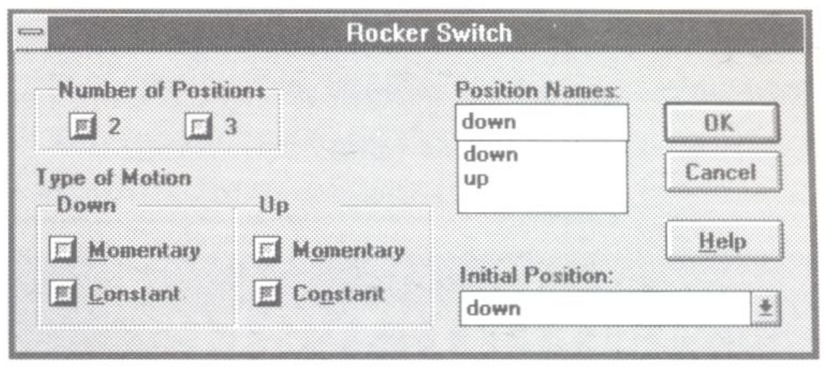


Figure 5: Configuring the switch.

Once you've created an object, whether it's a switch, lamp, pump, tank or anything else, you just double-click on it to edit it. The resulting dialog boxes from double-clicking the rocker switch are as shown in Figures 4 and 5. These dialog boxes allow you to set default properties and behaviors. I've chosen to name my two position switches "WallSwitch" and to name the positions "up" and "down." The positions could have been "on" and "off" or "hot" and "cold" if I wanted. Just like Authorware icons and variables, judicious naming of positions will make your logic clearer.

All of the other objects in Rapid, from primitive shapes to animated flow pipes, have similar property dialog boxes. Furthermore, you can customize the look of any object with your own bitmaps. Those who want to roll up their sleeves can even create new objects using the Object Editor. To keep this example simple, I used only built-in objects.

Modes and Transitions

After completing the layout, I used the Mode Tree, to create modes or states. From the Mode Tree, I created a

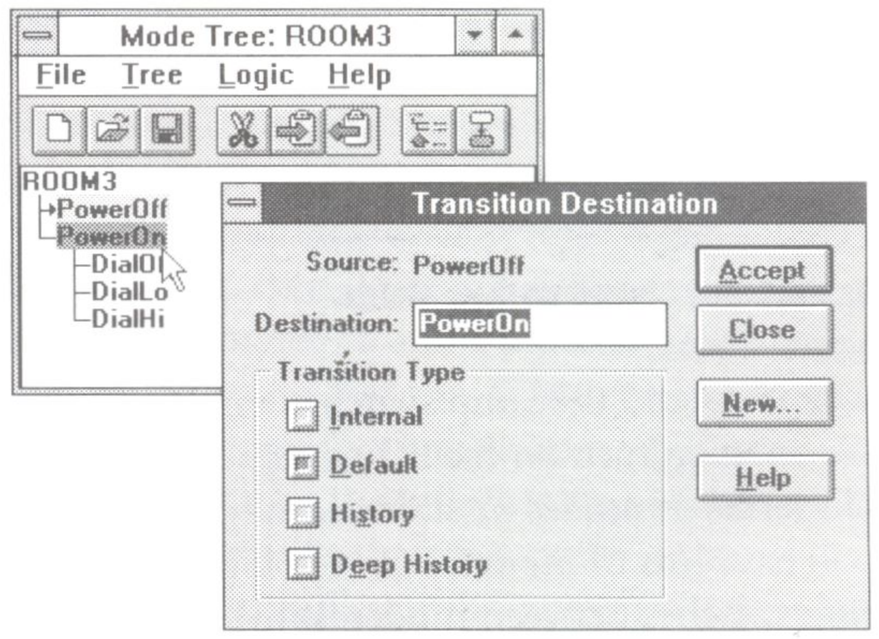


Figure 6: Creating a transition from the Mode Tree.

transition by selecting one mode and then Control-clicking on another mode to complete the dialog box shown in Figure 6.

With the destination set, I switched over to the mode browser and began making a new trigger. To build the conditions for the trigger, I just selected options from the Logic Editor, as shown in Figure 7. The Logic Editor lets you browse through all the objects you've created, as well as built-in system objects. Only valid properties and functions are displayed, making it virtually foolproof—no syntax errors!

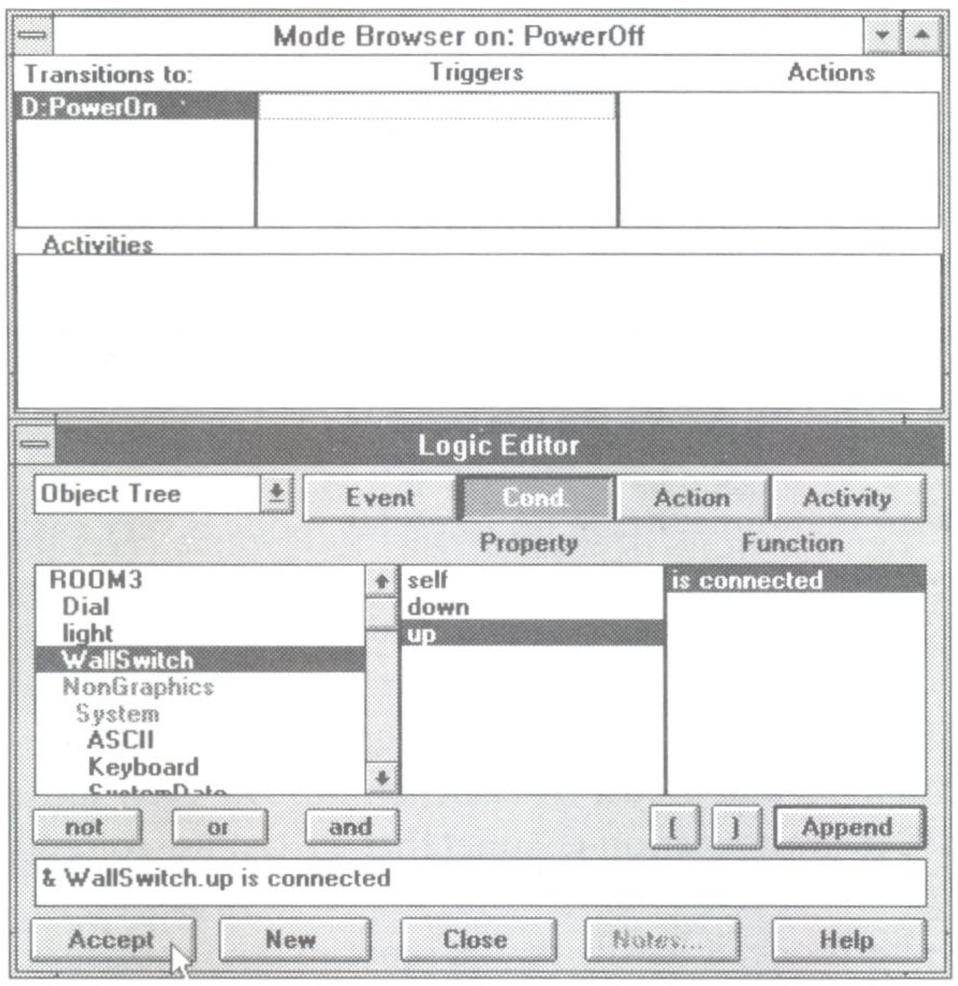


Figure 7: Creating a trigger for the transition.

Making Things Happen

What's the point of creating all these modes unless something happens when you get there? To make things happen, you arrange for activities to occur when entering or exiting a mode. You can also have activities that repeat as long as you are in a mode. Adding activities is just like adding transition triggers. First, use the Tree Chart to select a mode. Then use the mode browser and Logic Editor to add an activity using the same click-and-add interface used for triggers. Figure 8 shows the activity for the DialHigh mode. Note that this activity happens once each time you enter the DialHigh mode. If an activity needs to occur repeatedly (for instance, to send messages that might trigger other transitions), then that activity can be a mode activity. Mode activities are somewhat similar to perpetual sounds in Authorware; they keep repeating until something causes them to stop.

To complete my simulation, I added transitions between the states, appropriate triggers (see Table 2), and

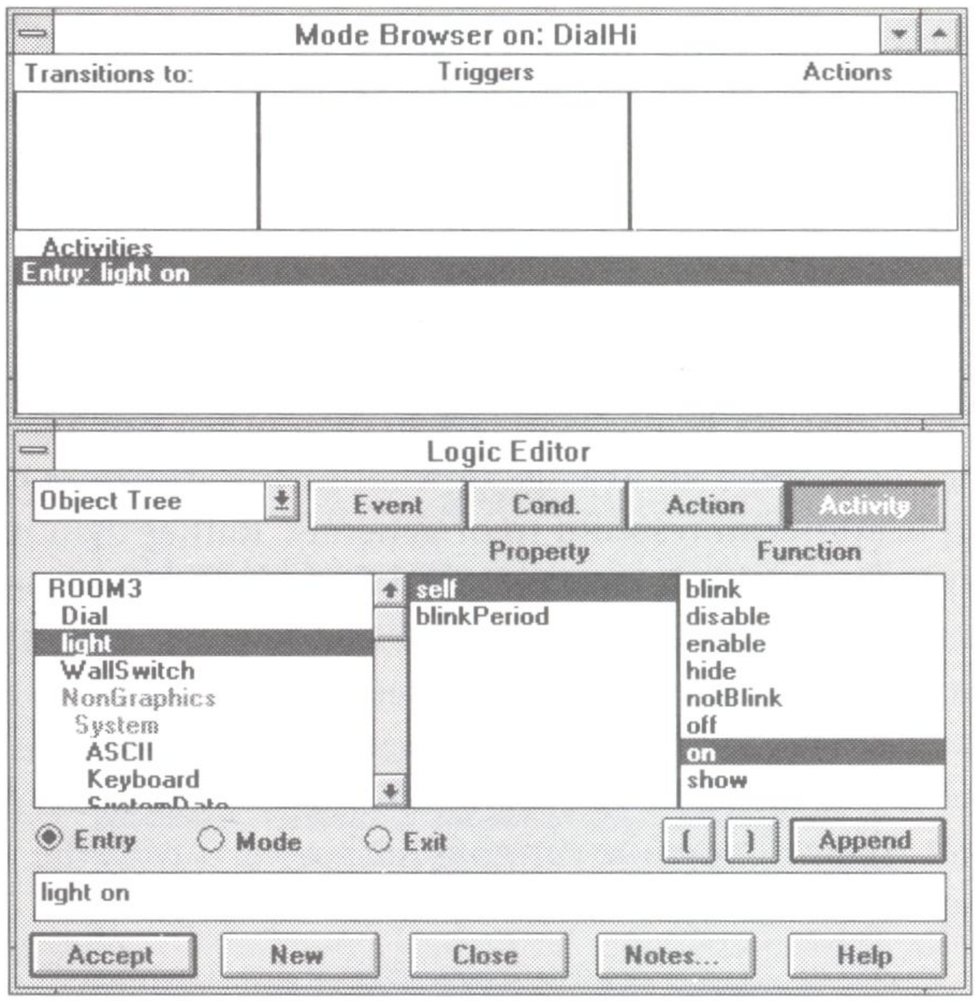


Figure 8: Creating an activity.

set entry activities for the modes that made the light blink (my "low" setting) or turn on ("high") or turn off. Running the simulation is simply a matter of clicking on the toolbar icon for the Prototyper. Voilà, my simulation is up and running, as shown in Figure 9.

The Prototyper's Option menu lets me select a Trace mode. The Tree Chart is dynamically updated in Trace mode to show transitions and active modes as I flip switches. As the name implies, Trace mode is great for debugging. Other capabilities of the Prototyper let you "record" simulations for later playback for demonstration or instruction, which of course leads us back to

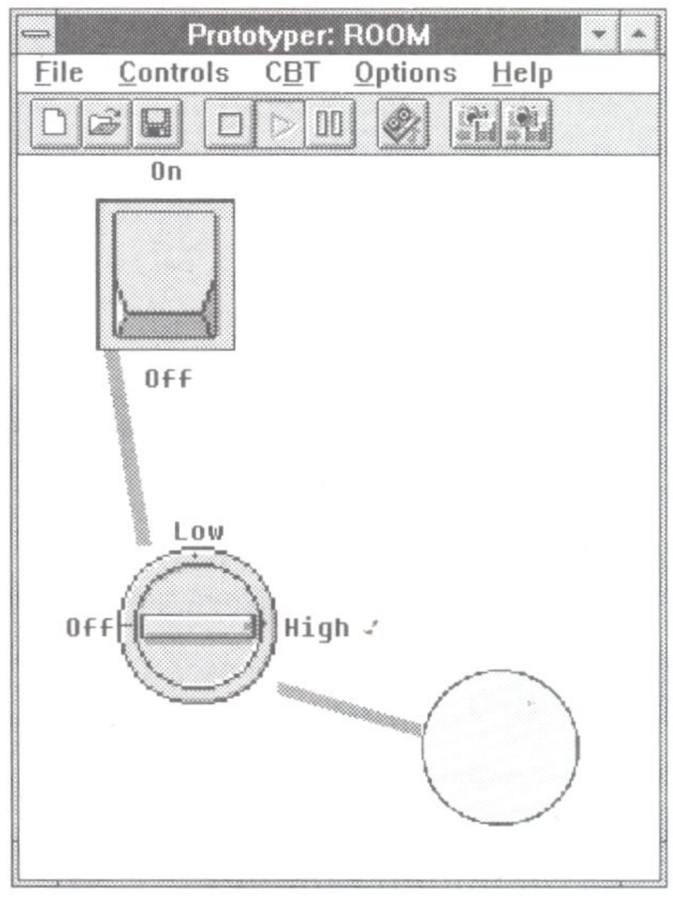


Figure 9: Running the simulation in the Prototyper.

Authorware. The Prototyper (and Rapid Reviewer, the Rapid ST equivalent of RunA3W) lets you run free-play simulations, but how does this help me with Authorware? Stay tuned for details.

Dynamic Duos and Dynamic Data Exchange (DDE)

Dynamic Data Exchange (DDE) is a messaging service that allows applications to talk to each other, provided they have some knowledge of what sort of messages each application can respond to or generate. Rather than using my tedious table lamp example, I'd like to present communication between Authorware and Rapid as something more interesting. So I'll use pieces of the Global Positioning Satellite (GPS) navigation demo available on the Rapid ST 3.0 demo CD-ROM. Figure 10 is taken from that application while running Authorware, with the Rapid GPS simulation overlaying the right side.

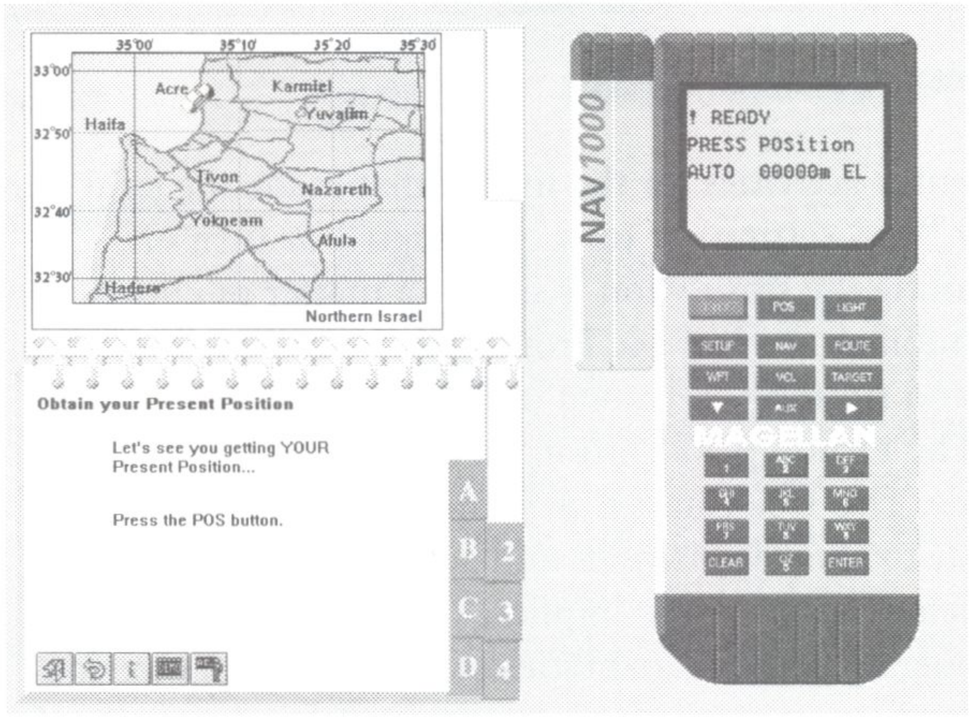


Figure 10: Running GPS in the Prototyper.

As its name implies, DDE is dynamic (rather than store-and-forward), so both programs must be ready to generate and receive messages. This is somewhat like making a phone call—you need to know who to call, and they have to be there to answer. Getting a DDE channel started is relatively easy using the RAPIDLNK.UCD file included with Rapid. (A UCD is an Authorware-specific DLL format that includes resources describing each function, making it easier for authors to use.) In fact, the DDE messaging is virtually transparent.

Before you can send or receive messages, you'll need to get Rapid going. To start Rapid from within Authorware 3.0 for Windows, load the appropriate function, as shown in Figure 11, and create a calculation icon like the one shown there.

Load Rapid & set up Window

```
– Hide Rapid if it's running
status:=ShowOrHidePrototyper(0)

– only load Rapid if necessary
if status=0 then LoadRapid(".", ".\\applics\\gps.rpd", UserName, "/h")

– position Rapid Window
SetPosition( 450, 20, -1, -1)

– make it a slave window (no borders)
SlaveOrReleasePrototyper(1)
```

Figure 11: Getting Rapid Reviewer running.

Start Free Play

```
– start DDE (for later use)
ddelink:=DDEInitialize( "GPS", "CBT")

– show rapid & make it run in free
– play (normal) mode at the top level
ShowOrHidePrototyper(1)
NormalEvents()
```

Figure 12: Opening DDE and starting free play.

With that done, firing up free-play mode is very easy, as shown in Figure 12.

That's it for free play. After running through that second calculation, Rapid will take care of all the simulation events. Whew, we did it: Rapid is running from within Authorware, and in just two calculations. We even started a DDE communication channel that we can use to send and receive specific information to and from Rapid. In free-play mode, it hardly makes sense to send messages back and forth, so we didn't really do anything more than start DDE. Fortunately for me, it won't take much more to get the Monitor mode going. The UCD file included with Rapid spares us the details of sending everything as DDE messages, but still allows detailed messaging if needed.

Big Brother Is Watching

Free play's first cousin is Monitor mode. When Rapid runs in Monitor mode, Authorware can watch but it can't trigger simulation events like a button press. The example I typically use is a complex Hewlett-Packard calculator. Bankers and other lenders often use preprogrammed functions to generate payment or amortization-table information. Some loans work on simple interest, while some, like car loans, are based on more complex calculations.

There are several ways to calculate the loan payments, but all of them should arrive at the same result. However, if someone is doing this all day, you might want to coach them on the fastest way to do it. Why not demonstrate that in Authorware and then fire up a Monitor mode simulation? Rapid can simulate the calculator, while sending messages to Authorware about which keys are pressed in

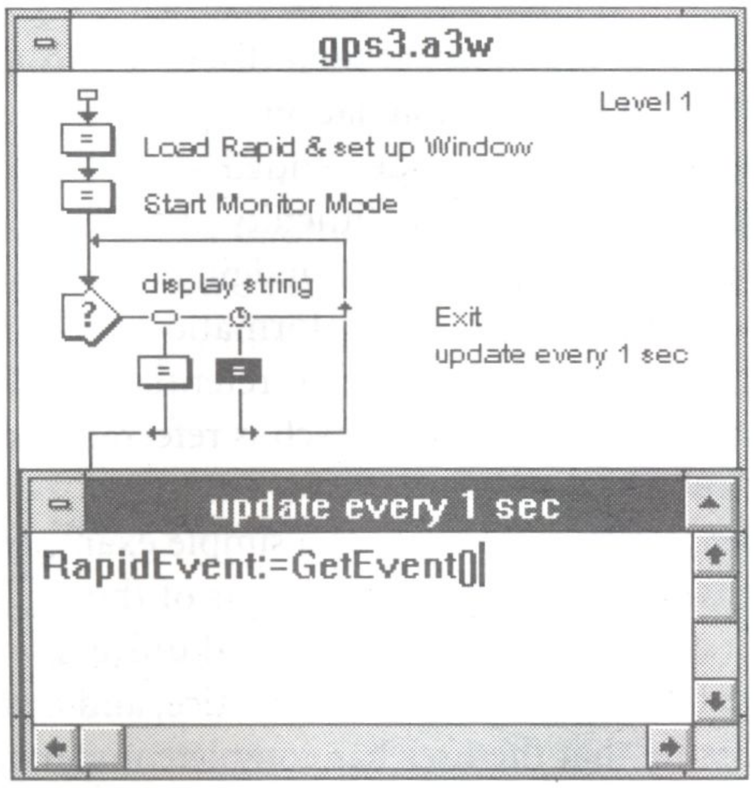


Figure 13: Monitoring the action.

what order. Authorware can use this information to let students calculate payments however they want, and then coach them afterward on how they might have done it more efficiently.

Monitor mode is handy whenever you don't need to watch *every* action, but you do want to know the end result or just keep tabs on what has happened. Starting Monitor mode is as easy as replacing the NormalEvents() call in Figure 12 with a call to MonitorEvents().

If you've used Authorware with other programs or any sort of polling loops, the rest is easy. Suppose, for instance, that we want to display the most recent Rapid event on screen once per second until the user clicks an

```
Start Re-route Mode
– start DDE (for later use)
ddelink:=DDEInitialize( "GPS", "CBT")

– show rapid & make it run in free
– play (normal) mode at the top level
ShowOrHidePrototyper(1)
RerouteEvents()
StayOnTop(1)
```

Figure 14: Startup modified for Reroute mode.

Exit button. Assuming that MonitorEvents() has been inserted at the proper location, it's as easy as embedding {RapidEvents} in an interaction icon and creating a Time Limit response with a repeating one-second trigger. Figure 13 shows an Authorware sample to do just that.

The Authorware string variable will end up holding cryptic messages like: "On $self out time ID." The format of these messages is: *<ObjectName> <ObjectProperty> <Event> <TimeStamp> <EventID>*. Knowing this, I can use the Authorware GetWord() function to extract object names or events. There are even slicker ways to get specific messages via DDE, by requesting the status of an object. This information can be used by Authorware to watch almost anything that happens within the simulation. Now I'm getting tempted to filter these events rather than just watch. Well, The rapidlnk UCD will let you do just that.

I know all kinds of useful reasons why you might want to filter events, such as giving diagnostic feedback, hints or

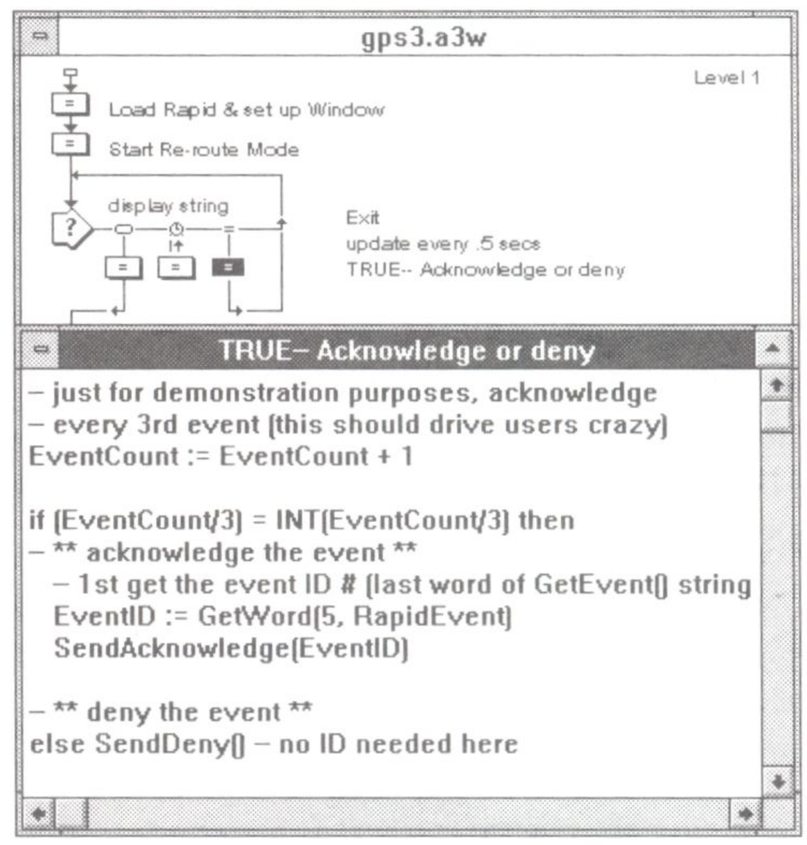

gps3.a3w
Level 1
Load Rapid & set up Window
Start Re-route Mode
display string
Exit
update every .5 secs
TRUE-- Acknowledge or deny
TRUE– Acknowledge or deny
– just for demonstration purposes, acknowledge
– every 3rd event (this should drive users crazy)
EventCount := EventCount + 1
if (EventCount/3) = INT(EventCount/3) then
– ** acknowledge the event **
– 1st get the event ID # (last word of GetEvent() string
EventID := GetWord(5, RapidEvent)
SendAcknowledge(EventID)
– ** deny the event **
else SendDeny() – no ID needed here

Figure 15: Filtering events.

even updating the training text (or displays) to reflect user selections in the simulation. But I wanted to do something silly, like make the buttons work only on the *third* time you press them, or maybe only every third thing you do will work. Fortunately for me (and unfortunately for the quality-assurance team I gave this to), it's easy to have Authorware make Rapid ignore two out of three events. All I did was modify the startup code to use RerouteEvents() (see Figure 14), and make a few changes to the interaction used in the previous examples.

To set things up so that only every third event passed through to Rapid, I created a counter (EventCounter) and increased it incrementally each time an event was received. Then I did a little math to make sure the counter is evenly divisible by three. Sending an acknowledge message back to Rapid with an Event ID causes it to process that event. Sending an Event ID requires some simple parsing of the event string to get the ID, but it assures that *only* the specified event is acted on by Rapid.

Of course, for the two times out of three that the counter is not evenly divisible by three, I send a deny message back to Rapid. In the Reroute mode, Rapid ignores events until it receives a message from Authorware telling it to process the event. This makes denial almost a default, so no Event ID is needed when I call the SendDeny() function.

Finally, to make my simulation more responsive, I decreased the length of the Time Limit response from one second to a half second. I also added {EventCounter} as an embedded variable so I could keep track of which event Rapid should respond to (e.g., 3, 6, 9, 12...). I put the acknowledge or deny logic in a conditional response and changed the branching of the Time Limit response to Continue so that my TRUE conditional gets triggered every time the Time Limit fires. Figure 15 shows the final results of my authoring.

That's it for Rapid and Authorware. If you would like more information about Rapid, please contact Emultek at one of the addresses shown below.

Emultek North America, 284 Racebrook Rd., Orange, CT 06477. (203) 891-8495. Fax (203) 891-8497. E-mail: 75052.2265@compuserve.com.

Emultek International, P.O. Box 45002, Jerusalem, Israel 91450. +972-2-870770. Fax: +972-2-870773. E-mail: 100274.2054@compuserve.com.

Tom King (SolisInc@aol.com) monitors minions, generates external events and practically lives at Solis Design Inc. in San Mateo, California.

Tips and Techniques

Judge for Yourself

Closed Instructional Strategies

By Jonathan Adams

One of the more frequently asked questions that come up when folks are considering software as an alternative to stand-up or real-life training is whether Director can perform answer judging—comparing what the user types in to what has been stored as the correct answer. Unlike Authorware, Director does not offer an extensive menu-driven environment for creating instructional software. For developers coming from authoring environments that do, this question is quite understandable. At first glance, Director might not appear to have this capability. In this article, I will review the basics of a universal, computer-based lesson design and provide a framework for answer-judging text entries.

Closed Instructional Strategies

In the December 1995 issue of *MUJ* I demonstrated an open instructional strategy by developing a message pad, an object that users can use at their discretion to record their experiences. As the designation suggests, open instructional strategies are open ended and user driven, factors that contribute to the difficulty of their development. On the other hand, answer judging infers that the user has acquired some sort of information and will be tested to measure their skill level or retention of information. In design circles, this approach is referred to as a closed instructional strategy.

Consider the game of chess as a simple example to contrast these two designs. The movements of the individual pieces on a chessboard are standard and universal. Their actions can be taught by drill and practice, and can be tested to ensure that the user has complete understanding of how each piece moves.

This type of information is ideal for closed instructional strategies. Playing a game of chess, however, requires a completely different set of skills. Once the basic openings are mastered, myriad strategies exist for interactions between the pieces. These strategies are open ended, which means they can be altered to fit new situations. Some may be better than others, but these strategies cannot be judged on a right-or-wrong basis. So it may be a relatively simple matter to develop a Director piece to instruct and test users' ability to remember and move chess pieces, but another matter to develop a working chess master to play against.

When answer judging is a requirement, the possibilities for data entry are limited. Closed instructional strategies frequently employed in computer-based instruction are checking multiple-choice options (pointing and clicking), positioning objects in sequence (dragging objects) and typing in answers (keying information).

Basic Model

To start, let's take a look at a typical flow chart of how answer judging of a text entry works. The user is given information (information presentation) and is required to

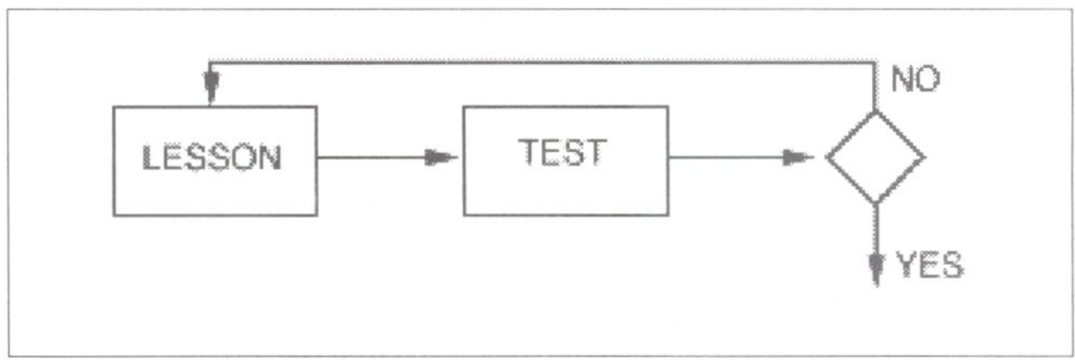


Figure 1: Universal lesson design.

provide information, which is compared to the right answer. This approach is a closed instructional strategy, since we are engaging the user in recognition or recall of information rather than experiential learning, as our

open-ended strategies might. Figure 1 shows a "universal" lesson design.

Answer judging then sends the user on to the next question or back for another try. To highlight this type of action, here's a simple script to check whether a text field has been left empty or starts with a space.

```
on mouseUp
     if the text of cast "Data" = EMPTY then
     alert "Please make an entry!"
   else
     set CheckChar = char 1 of field "Data"
     if checkchar = " " then
       alert "Please remove spaces!"
       hilite char 1 of field "Data"
     end if
   end if
end
```

textEntry	Stores data from the entry text field
answer:	Holds the correct answer
userResponseLength	Holds the number of characters in the entry
answerLength	Holds the number of characters in the answer
firstChar	The first text character of the string
lastChar	The last text character of the string
responseChars	Holds individual letters of the answer field
answerCHars	Holds individual letters of the correct answer

To extend this simple model, let's look at an engine for comparing a chunk of text given by a user to a predetermined correct answer. In addition to making this comparison, this handler will also indicate where the mismatch occurs by highlighting the error. (More intrepid developers should refer to Kurt Cagle's "Truly Vexing Text Tricks" article in *MUJ,* July 1995, or Hyperstand, www.hyperstand.com, for information on how to check for case-sensitive mismatches.)

Checking Entries

We need to use several local variables in order to check the string entries for correctness. Listed below are the names and descriptions of local variables in the answer-judging handler.

First, set up all of the local variables needed to test the length of the string entry, measure the number of characters in the string and compare the entry to the correct answer. The second step is to take one letter from each string and compare the characters, one by one. If a mismatch is found, then highlight the error.

```
  on JudgeText
     set textEntry = field "UserResponse"
     set answer = "Test run"
     set userResponseLength = the number of chars
in TextEntry
     set answerLength = the number of chars in
Answer
     set firstChar = 1
     set lastChar = UserResponseLength

     repeat while char firstChar of textEntry=" "
       put firstChar + 1 into firstChar
     end repeat

     repeat while char lastChar of textEntry=" "
       put lastChar - 1 into lastChar
       if lastChar = 0 then
         exit repeat
       end if
     end repeat
```

The following nested loop compares the two strings letter by letter and inserts a space for the selEnd text property. The selStart has to be set to where the comparison does not match, but at minus 1 in order to get the highlight to include the errant letter.

```
     repeat with i = 1 to AnswerLength
       set responseChars = char i of textEntry
       set answerChars = char i of answer
       if i > lastChar then exit repeat
       if responseChars <> answerChars then
         set highLightBegin = i-1
         if userResponseLength < highLightBegin +
1 then put " " after field "userResponse"
         set the selEnd to userResponseLength + 1
         set the selStart to highLightBegin +
firstChar - 1
         exit repeat
       end if
       end repeat
  end JudgeText
```

This example clearly shows the power and versatility of Director when developing a closed instructional strategy involving answer judging. While closed instructional strategies typically address the basics, their use is not limited to education or training projects, nor are they limited to simple answer-judging design for pointing and clicking and dragging objects. If you look at marketing, education, training and games in terms of open and closed strategies and keep score, you may discover a missing element to develop for your personal Director library files.

Jonathan Adams (PilotMedia@aol.com) is principal consultant with Pilot Media Inc. He teaches Director and Scripting with Lingo in the University of Massachusetts multimedia certificate program.

Version Watch — January 1996

Product	Platform	Version	Price
Application Software			
Authorware (Includes Director Studio bundle)	Mac	3	$4,995
Authorware (Includes Director Studio bundle)	Win	3	$4,995
Action!	Mac	1.0.4	$199
Action!	Win	3.0	$199
Action! CD-ROM Edition (Action! SoundEdit Pro, ClipMedia 3)	Mac	—	$299
Action! CD-ROM Edition (Action! Turtle Beach's Wave Tools, ClipMedia 2)	Win	—	$299
Deck II	Mac	2.5	$399
Director ($29 upgrade for registered 4.0 Mac users)	Mac	4.0.4	$1,195
	Win	4.0.4	$1,195
Fauve xRes	Mac	1.2	$799
Fontographer	Mac	4.1	$495
Fontographer	Win	3.5	$495
FreeHand	Mac	5.5	$595
FreeHand	Win	5.0	$595
MacroModel	Mac	1.5	$895
MacroModel (includes Pixar's MacRenderMan)	Mac	1.5	$995
MacroModel (includes Macromedia Three-D)	Mac	1.5	$995
MacroModel (includes Pixar's Windows RenderMan)	Win	1.5	$995
SoundEdit 16 (upgrades from earlier versions of SoundEdit 16, $129)	Mac	2.0	$399
X-Objects Developer's Toolkit (Windows version is included with Director for Windows 4.0)	Mac	N/A	$50

Director Studio bundle, $1,995 for Macintosh or Windows, includes Director, MacroModel, SoundEdit 16 (Mac), SoundForge (Win), and Adobe Premiere.

Graphic Design Studio bundle, $995 for Macintosh or Windows, includes FreeHand, MacroModel, Fontographer, Pixar RenderMan and Fractal Design Painter.

Product	Platform	Version	Price
ClipMedia			
ClipMedia 1: Business & Technology	Mac	2.0	$395
ClipMedia 2: Industry at Work	Mac	2.0	$295
ClipMedia 3: Perfect Presentations	Mac	2.0	$195
ClipMedia 1: Business & Technology	Win	2.0	$395
ClipMedia 2: Industry at Work	Win	2.0	$295
ClipMedia 3: Perfect Presentations	Win	2.0	$195

Coming Soon in the MUJ

- Tom King looks at Designer's Edge, Allen Communication's new pre-authoring tool for Authorware, Quest and IconAuthor users.
- A sneak peek at Macromedia's first rev of xRes.
- Tony Bové shows you how to master CD-ROM load time and digital video space with Director.
- Jonathan Adams teaches you how to make your buttons confirm to Apple standards.
- Vector graphics on the Web with Shockwave for FreeHand.